MODÜL 9

SÖZLÜKLER VE DEMETLER

Şekil 9.1: *Bölümle ilgili örnek uygulamalara karekoddan ulaşabilirsiniz.*

Sözlükler, gerçek hayattaki sözlükler gibi davranan bir veri tipi olarak bilinmektedir. Python ile JSON formatında verilerde İsim Değer Çifti Koleksiyonu sözlük yapısıyla birebir uyumludur. Bundan dolayı sözlük yapısı JSON verilerinde kullanılmaktadır. Sözlüğün içindeki her bir eleman indeks olarak değil, anahtar (key) ve değer (value) olarak tutulur. Anahtar değer çiftleri virgülle ayrılarak bir öğeyi oluştururlar. Değer kısmı bütün veri türünü içerebilir fakat anahtar kısmı sadece string ve int tipinde olabilir. Sözlükler sırasız bir öge koleksiyonudur. Anahtar ve değer yapısında veri kümeleri oluştururlar. Değerler herhangi bir veri türüne ait olabilir ve tekrarlanabilir fakat anahtarlar kesinlikle benzersiz olmalıdır.

9.1. Sözlük Oluşturma

Sözlük oluştururken dikkat edilmesi gerekenler:

- İki noktayı ve virgülü nereye koyduğumuza,
- Ögelerimizi tanımlarken ayraç parantez "()" yerine Küme Parantezi "{}" kullandığınıza,
- Anahtar-değer ilişkisine,
- Anahtar değeri tanımlarken tırnak işareti "" kullandığınıza dikkat ediniz.

Aşağıdaki örnekte, süslü parantez ve iki nokta {:} ile anahtar değerlerinizi yerleştirerek kişi bilgilerini içeren bir sözlük oluşturunuz.

```
kisisel_bilgiler={"ad":"ali","telefon":"05554442211","e_posta":"ali@abc.
com","ülke":"Türkiye","il":"istanbul","ilçe":"beşiktaş"}
print(kisisel_bilgiler)
{'ad': 'ali', 'telefon': '05554442211', 'e_posta': 'ali@abc.com', 'ülke': 'Türkiye',
'il': 'istanbul', 'ilçe': 'beşiktaş'}
```

Şimdi de key tiplerine göre sözlükler oluşturup ekrana yazdırınız.

dict( ) fonksiyonu kullanarak sözlük oluşturabilirsiniz. Örnek 1'de görüldüğü gibi sozluk4 adında 2 adet ögesi olan sözlük oluşturulmuştur.

Örnek 1

```
# bos bir sozluk
sozluk1 = {}
print(sozluk1)
# integer veri tipi oluşturulan keyler
sozluk2= {1: 'adana', 2: 'adıyaman'}
print(sozluk2)
# string ve integer veri tipi ile oluşan keyler
sozluk3 = {'isim': 'ali', 1: [5, 4, 3]}
print(sozluk3)
# dict fonksiyonu
sozluk4 = dict({1:'erik', 2:'ayva'})
print(sozluk4)
{}
{1: 'adana', 2: 'adıyaman'}
{'isim': 'ali', 1: [5, 4, 3]}
{1: 'erik', 2: 'ayva'}
```

Örnek 2'de 3 adet anahtara sahip sözlük oluşturularak listelenmiştir.

Örnek 2

```
sozluk = {"bilgisayar" : "computer","sarı" : "yellow","masa" : "chair"}
print(sozluk)
{'bilgisayar': 'computer', 'sarı': 'yellow', 'masa': 'chair'}
```

9.2. Sözlük Anahtar ve Değerlerine Erişim

Sözlük anahtar ve değerlerini listelemek için keys ve values komutları kullanılır.

Örnek 3

#keys( ) metodu sözlükteki anahtarları yazdırır.

```
iller={ "konya" :"42","istanbul"   :"34","ankara" :"06" }
print ( iller.keys())
#values()metodu  sözlükteki değerleri bize yazdırır.
print (iller.values())
dict_keys(['konya', 'istanbul', 'ankara'])
dict_values(['42', '34', '06'])
```

Örnek 3'te iller sözlüğünde önce anahtarlar sonra değerleri listelenmiştir. Anahtarlar 'konya', 'istanbul', 'ankara' ve değerleri ise '42', '34', '06' şeklinde vermiştir.

9.3. Sözlüklerde Eleman Seçme, Silme, Ekleme, Değiştirme

9.3.1. Sözlükte Eleman Seçme İşlemleri

Örnek 4'te meyveler adında sözlük oluşturulmuştur. Bu sözlükte adı, türü ve kg adında 3 adet anahtar ve bu anahtarlara ait değerleri mevcuttur. Araçlar sözlüğünde adı ve kg anahtarları ekrana yazdırılarak eleman seçme işlemi yapılmıştır.

Örnek 4

```
meyveler={"Adı": "Portakal","Türü": "Turunçgiller","Kg": 20}
print(meyveler["Adı"])
print(meyveler["Kg"])
Portakal
20
```

Örnek 5

```
kisi_bilgileri = {'ad':'ali','yas':40,'memleketi':'konya'}
print(kisi_bilgileri)
# deger değiştirme
kisi_bilgileri['yas'] =45
    print(kisi_bilgileri)
    # deger ekleme
   kisi_bilgileri['adres'] = 'nişantaşı'
   print(kisi_bilgileri)
{'ad': 'ali', 'yas': 40, 'memleketi': 'konya'}    1.çıktı
{'ad': 'ali', 'yas': 45, 'memleketi': 'konya'}    2.çıktı
{'ad': 'ali', 'yas': 45, 'memleketi': 'konya', 'adres': 'nişantaşı'}   3.çıktı
```

Örnek 5'te kişi bilgileri adında sözlük oluşturulmuştur. Bu sözlükte ad, yas, memleketi adında 3 anahtar ve değerleri tanımlanmıştır. Bu değerler 1.çıktıda ekrana yazdırılmıştır. Daha sonra yas değeri 45 olarak değiştirilmiştir. 2. çıktıda tüm değerler ekrana yazdırılarak yas değerinin 45 olarak değiştirildiği görülmüştür. 3. çıktıda ise adres adında anahtar ve değeri eklenerek listelenmiştir.

9.3.2. Sözlükte Eleman Silme İşlemleri

pop( ) metodu, öğeyi belirtilen anahtar adıyla kaldırmak için kullanılır.

Örnek 6

```
kisi_bilgileri= {"Adı": "Ekrem","Soyadı": "Yıldırır","Yaşı": 40 } print(kisi_
bilgileri)
kisi_bilgileri.pop("Yaşı")
print(kisi_bilgileri)
{'Adı': 'Ekrem', 'Soyadı': 'Yıldırır', 'Yaşı': 40}
{'Adı': 'Ekrem', 'Soyadı': 'Yıldırır'}
```

Örnek 6'yı incelediğimizde Yaşı anahtarı pop metodu ile kaldırılmıştır. Kisi_bilgileri sözlüğü listelenerek sadece Adı ve Soyadı anahtarlarının olduğu görülmüştür.

popitem( ) metodu, eklenen son öğeyi kaldırmak için kullanılır. Örnek 7'de popitem metodu kullanılmıştır. Bu metot kullanılarak, kisi_bilgileri adlı sözlükte son anahtar olan Yaşı kaldırılmıştır.

Örnek 7

```
kisi_bilgileri= {
"Adı": "Ekrem",
"Soyadı": "Yıldırır",
"Yaşı": 40
}
print(kisi_bilgileri)
kisi_bilgileri.popitem()
print(kisi_bilgileri)
{'Adı': 'Ekrem', 'Soyadı': 'Yıldırır', 'Yaşı': 40}
{'Adı': 'Ekrem', 'Soyadı': 'Yıldırır'}
```

del (parametre) anahtar sözcüğü, belirtilen anahtar adına sahip öğeyi kaldırmak için kullanılmaktadır. Örnek 8'de del komutu ile Yaşı adlı anahtar değere ait 40 değeri silinmiştir.

Örnek 8

```
kisi_bilgileri= {
"Adı": "Ekrem",
"Soyadı": "Yıldırır",
"Yaşı": 40
}
del kisi_bilgileri["Yaşı"]
print(kisi_bilgileri)

{'Adı': 'Ekrem', 'Soyadı': 'Yıldırır'}
```

clear( ) anahtar sözcüğü sözlüğü boşaltırken, **del** anahtar sözcüğü ise ayrıca sözlüğü tamamen silmektedir. Aşağıdaki örnekte clear( ) anahtar sözcüğü kullanılmış ve sözlüğün içi boşaltılarak ekrana yazdırılmıştır.

```
sozluk ={"black":"siyah",   "green":"yeşil",   "white":"beyaz",}
print(sozluk)
sozluk.clear()
print(sozluk)
{'black': 'siyah', 'green': 'yeşil', 'white': 'beyaz'}
{}
```

Örnek 9

```
kisi_bilgileri= {"Adı": "Ekrem","Soyadı": "Yıldırır","Yaşı": 40}
kisi_bilgileri.clear()
print(kisi_bilgileri)
del kisi_bilgileri
print(kisi_bilgileri)

{}
NameError Traceback (most recent call last)
 HYPERLINK "https://localhost:8080/" <ipython-input-9-c04608b44e6f> in <module>()
      3 print(kisi_bilgileri)
      4 del kisi_bilgileri
----> 5 print(kisi_bilgileri)

NameError: name 'kisi_bilgileri' is not defined
```

Örnek 9'da kisi_bilgileri sözlüğünde clear( ) metodu ile içi boşaltılmış. Daha sonra del( ) metodu ile kisi_bilgileri sözlüğü silinmiştir. Ekrana listele işlemi yapıldığında kisi_bilgileri adlı sözlük olmadığı için hata mesajı ile karşılaşılmıştır.

9.3.3. Sözlüğe Eleman Ekleme İşlemleri

Sözlüklere veri ekleme işlemi yapılabilmektedir. Yöntem sözlükadı["anahtar"]="deger" şeklinde olmaktadır. Örnek 10'da sözlükte önce tek öğesi bulunmaktadır. İkinci ve üçüncü öge eklenerek listeleme işlemi yapılmıştır.

Örnek 10

```
sozluk={"adi":"sami", }
print(sozluk)
sozluk['soyadi']='yılmaz'
print(sozluk)
sozluk['yasi']=40
print (sozluk)
{'adi': 'sami'}
{'adi': 'sami', 'soyadi': 'yılmaz'}
{'adi': 'sami', 'soyadi': 'yılmaz', 'yasi': 40}
```

9.3.4. Sözlükte Eleman Değiştirme İşlemleri

Sözlükte eleman değiştirmek için anahtarı kullanarak güncelleme işlemi yapılmıştır.

Kullanımı:

sözlük adı['anahtar adı']=yeni değer şeklinde yapılmıştır.

Örnek 11

```
sozluk = {'isim':'ahmet','yas':40}
# deger değiştirme
sozluk['yas'] = 45
print(sozluk['yas'])
45
```

Örnek 11'de olduğu gibi sözlüğün yas adlı anahtarının değeri 40 iken 45 olarak değiştirilmiştir.

9.4. Sözlükler Üzerinde Gezinme

Python'da sözlükler öğelerine erişmek için anahtarlarını (KEY) kullanırlar ve değerler bu indexler üzerinden gösterilir. Anahtarlar köşeli parantez içerisinde ya da get( ) fonksiyonu ile kullanılır. get( ) fonksiyonunda anahtar bulunamazsa KeyError yerine "None" döndürür.

Örnek 12

```
sozluk = {'ad': 'ali', 1: [5, 4]}
print(sozluk['ad'])
print(sozluk.get(1))
ali
[5, 4]
```

Örnek 12'de sozluk adlı sözlükte ad ve 1 adında anahtarlar ve bu anahtarlara bağlı ali ve 5,4 değerleri bulunmaktadır. Ad ve 1 anahtarlarını kullanılarak değerleri ekranda listelenmiştir.

Örnek 13'te sozluk1 adında sözlük oluşturulmuş ve bir ve iki anahtarları kullanarak değerlerine erişilmiştir.

Örnek 13

```
sozluk1 = {"sıfır":0,"bir":1,"iki":2,"üç":3}
print(sozluk1)
# "bir" anahtarına karşılık gelen değeri buluyoruz.
print(sozluk1 ["bir"])
# "iki" anahtarına karşılık gelen değeri buluyoruz.
print(sozluk1["iki"])
{'sıfır': 0, 'bir': 1, 'iki': 2, 'üç': 3}
1
2
```

Örnek 14

```
deneme = {"sıfır":0,"bir":1,"iki":2,"üç":3}
print(deneme["on"])
Traceback (most recent call last):
File "<ipython-input-22-e25f3b4387f3>", line 2, in <module>
print(deneme["on"])
KeyError: 'on'
```

Örnek 14'te sözlükte olmayan bir anahtar girildiğinde ekrana KeyError:'on' diye key hatası mesajı verecektir.

9.5. Demet (tuple) Oluşturma ve Eleman İşlemleri

Demetler veri ekleme ve çıkarmanın yapılamadığı veri yapılarıdır. Bu değiştirilemez özellikleri dışında listelere benzemektedirler. Demetler ekleme çıkarma gibi işlemlerle uğraşmadığı için daha hızlı çalışır. Eklenen verilerin program boyunca değiştirilmesinin istenmediği durumlarda kullanılmaktadır.

Boş bir demet oluşturma:

Örnek 15

```
demet=()
print(demet)
()
```

Örnek 15'te boş bir demet oluşturularak, ekrana listelenmiştir.

İndis içeren demet oluşturma:

Örnek 16

```
demet = ("Python","Java",2020,"JavaScript")
print(demet)
('Python', 'Java', 2020, 'JavaScript')
```

Örnek 16'da 4 ögesi bulunan bir demet oluşturularak ekrana yazdırılmıştır.

Tek elemanlı demet oluşturma:

Tek nesneli bir demet oluşturmak için nesneden sonra virgül konulması gerekmektedir. Örnek 17'de buna yönelik bir uygulamadır.

Örnek 17

```
meyveler =("erik",)
print(meyveler)
('erik',)
```

NOT

Eğer virgül koymazsanız bu değişken tipi demet değil string olarak sistemde yer alacaktır.

Demetler de sözlükler gibi indis olarak sıfırdan başlar. Örnek 18'de 6 elemanlı bir demet oluşturulmuştur. 1. İndise ulaşmak için demetin indis numarası kapalı parantez arasında girilerek ekrana yazdırılmıştır. print(demet[-1]) ise demet'in en son elemanı bize vermektedir. print(demet[2:])'da 2 numaralı indis dâhil iki numaradan sonraki tüm indislere ait elemanları listelemek için kullanılır.

Örnek 18

```
demet = (5,10,15,20,25,30)
# 1. indise ulaşma
print(demet[0])
# 3. indise ulaşma
print(demet[2])
print(demet[-1])
print(demet[2:])
5
15
30
(15, 20, 25, 30)
```

Aynı zamanda bir demet tuple( ) fonksiyonu ile de oluşturulabilir ve constructor yöntemiyle başlangıçta elemanları ayarlanabilir. Örnek 19'da tuple fonksiyonu ile demet oluşturulmuş ve öğeleri ekrana yazdırılmıştır.

Örnek 19

```
demet = tuple(("Ankara","İstanbul","Kayseri"))
print(demet)
('Ankara', 'İstanbul', 'Kayseri')
```

Örnek 20'de sayılar ve harfler diye iki adet demet tanımlanarak elemanları girilmiştir. Sayılar ve harfler demetleri toplanarak yeni_demet adlı demete aktarılmıştır. Elemanları ekrana yazdırıldı. Bu şekilde iki demet birleştirilmiştir.

Örnek 20

```
sayılar = (0,1,2,3,4,5,6,7,8,9,) #bir demet tanımladım
harfler = ("a","b","c","d","e") #ikinci bir demet tanımladım
yeni_demet = sayılar + harfler #tanımladığım demetleri topladım
print(yeni_demet) #yeni demeti ekrana yazdırdım
(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 'a', 'b', 'c', 'd', 'e')
```

Örnek 21'de olduğu gibi if koşul yapısı ile demetlerin içerisinde aranılan elemanların olup olmadığını kontrol edilmiştir. Gelen mantıksal cevabı ise ekrana true ya da false olarak yazdırılmıştır. "a" harfi harfler adlı demette olduğu için ekrana true olarak mesaj yazdırılmıştır.

Örnek 21

```
harfler = ("a","b","c","d","e")
if "a" in harfler: #True çıktısı verir
print(True)
else:
print(False)
True
```

Demet veri yapılarında elemanlar iki parantez arasına yazılır lakin bu parantez zorunlu değildir. Yine aynı şekilde liste veri yapılarında olduğu gibi demetlerde de dilimleme **(sliceable)** ve iç içe geçirilme **(nestable)** yapılabilir.

Örnek 22'de iç içe demetlere ait uygulama olup listeleme işlemi yapılmıştır. d adında demet oluşturulmuş ve iç içe demet yapısı olarak 2. indis numarasındaki ait 0. indis elemanına ve 3. indis numarasındaki 1. indis numarasına ait elemana ulaşılmıştır.

Örnek 22

```
d=((1,2),(3,4),(5,6),(7,8))
print(d[2][0])
print(d[3][1])
5
8
```

9.6. Demetlerin Temel Metotları

index metodu demette yer alan ögenin sırasını bulmaya yardımcı olur. Örnek 23'te demet adında demet oluşturulmuştur. 4 ögeden oluşan demette index metodu kullanılarak zambak adlı ögenin 2. sırada olduğunu görülmüştür.

Örnek 23

```
# Demet oluşturalım.
demet = ("lale","kardelen","zambak","papatya")
# "zambak" elemanının indeksini buluyoruz.
print(demet.index("zambak"))
2
```

Count metodu tuple içinde aynı elemandan kaç adet olduğunu bulunmasına yardımcı olur. Java adlı ögenin Örnek 24'te 3 adet olduğu görülmektedir.

Örnek 24

```
demet = ("Python","Java","C#","Delphi","C++","Java","Java")
print(demet.count("Java"))
3
```

Liste tipindeki veriyi tuple'a dönüştürmeyi değişkene aktarıp yapılabilir. Örnek 25'te aylar adlı liste demet adlı değişkene aktarılarak, demet dönüşümü yapılarak, ekrana yazdırılmıştır.

Örnek 25

```
aylar =["ocak","şubat","mart","nisan","mayıs","haziran"]
demet = tuple(aylar)
print(demet)
('ocak', 'şubat', 'mart', 'nisan', 'mayıs', 'haziran')
```

9.7. Demetler Üzerinde Gezinme

Demetler üzerinde gezinme işlemleri için for döngüsü kullanılır. Bu şekilde tüm öğeleri aynı anda listelenebilir. Örnek 26'da deneme adlı demet tanımlanarak içine 7 adet öge girilmiştir. Bu ögeler ise for döngüsü ile ekrana ögeleri alt alta yazdırılmıştır.

Örnek 26

```
deneme= (1,2,3,4,5,6,7)
for eleman in deneme:
print(eleman)
1
2
3
4
5
6
7
```

Örnek 27'de demette sorted( ) metodu ile ögeleri harflere göre sıralama yapılmıştır.

Örnek 27

```
demet =("lale","kardelen","zambak","papatya")
print(sorted(demet))
['kardelen', 'lale', 'papatya', 'zambak']
```

9.8. Bölüm Sonu Örnekleri

1. 5 ve 6 elemanlı olacak şekilde iki adet demet tanımlayınız. Bu iki adet demeti önce birleştirip sonra eleman sayılarını bulan programı yapınız.

2. `Sayilar=(20,24,25,79,40,39,50)` demet olarak tanımlanmıştır. Bu demet'te 5'e bölünenleri ekrana yazdıran programı yapınız.

3. Aşağıdaki kod çalıştırıldığında ekran çıktısı nedir?

```
demet = ("hasan","ali","c","mehmet","deniz","f","fatma")
yenidemet = demet[3:5]
print(yenidemet)
```

4. `Uygulama=("ali","veli","ayşe","Fatma","Hayriye","ali","deniz")` şeklinde tanımlanmış bir demet'te ali adlı öğenin kaç adet olduğunu bulunuz.

5. Aşağıdaki kod çalıştırıldığında ekran çıktısı nedir?

```
sozluk = {'renk': 'mavi', 'kıyafet': 'pantolon', 'beden': 'M'}
for anahtar in sozluk:
print(anahtar)
```

6. Aşağıdaki kod çalıştırıldığında ekran çıktısı nedir?

```
sozluk = {'renk': 'mavi', 'kıyafet': 'pantolon', 'beden': 'M'}
for anahtar in sozluk:
print(anahtar,sozluk[anahtar])
sozluk_bilesenleri=sozluk.items()
for bilesen in sozluk_bilesenleri :
print(bilesen)
print(type(bilesen))# demet veri tipinde oluyor
...
```

Cevaplar

1.
```
aylar =("ocak","şubat","mart","nisan","mayıs","haziran")
mevsimler = ("kış","ilkbahar","yaz","sonbahar")
yeni_demet=tuple(mevsimler+aylar)
print(len(yeni_demet))
10
```

2.
```
sayilar=(20,24,25,79,40,39,50)
for meyve in sayilar:
if meyve%5 ==0:
print(meyve)
20
25
40
50
```

3.
```
demet = ("hasan","ali","c","mehmet","deniz","f","fatma")
yenidemet = demet[3:5]
print(yenidemet)
('mehmet', 'deniz')
```

4.
```
uygulama=("ali","veli","ayşe","Fatma","Hayriye","ali","deniz")
print(uygulama.count("ali"))
2
```

5.
```
sozluk = {'renk': 'mavi', 'kıyafet': 'pantolon', 'beden': 'M'}
for anahtar in sozluk:
print(anahtar)
```
```
renk
kıyafet
beden
```

6.
```
sozluk = {'renk': 'mavi', 'kıyafet': 'pantolon', 'beden': 'M'}
for anahtar in sozluk:
print(anahtar,sozluk[anahtar])
sozluk_bilesenleri=sozluk.items()
for bilesen in sozluk_bilesenleri :
print(bilesen)
```
```
renk mavi
kıyafet pantolon
beden M
('renk', 'mavi')
<class 'tuple'>
('kıyafet', 'pantolon')
<class 'tuple'>
('beden', 'M')
<class 'tuple'>
```